解説

素因数分解技術の進展
― RSA-768 の分解達成への道のり ―

青木 和麻呂 日本電信電話（株）

2009 年 12 月 12 日（日本時間では 13 日），RSA-768 と呼ばれる 768 ビットの合成数が素因数分解された．特別な形の合成数では以前に，より大きいものが素因数分解されているが，RSA 公開鍵暗号に利用される大きな 2 素数の積の素因数分解に対しては，世界記録更新である．この記録は瑞，日，独，仏，蘭の 5 カ国の機関の研究者の共同作業の成果であり，筆者もこの作業に参加した．この素因数分解では，一般数体篩法（いっぱんすうたいふるいほう；GNFS）と呼ばれる代数的整数論を応用した方法が使われた．数体篩法を実際に実現するためには，概念的なアルゴリズムばかりではなく，細部の実装や計算資源の配分などが効率の重要な鍵を握っている．本稿では，数体篩法の概略を説明し，各ステップでの実装や計算機運用の問題点を紹介する．最後に，従来より RSA 公開鍵暗号で 1024 ビットの公開鍵の新規利用は推奨できないことが指摘されていたが，今回 768 ビットの合成数が現実に素因数分解されたことはこの指摘を補強するものであることを付け加えておく．

結果の概要

素因数分解問題とは，与えられた合成数 N に対し，それを素数の積で書き表す問題である．この問題は特殊な場合を除き，N が大きくなればなるほど困難な問題となる．紀元前より多くの数学者により研究され，巨大数の素因数分解競争が行われてきた．前世紀にコンピュータが実用化されたことにより競争が加速され，さらに安全性が素因数分解問題の困難性に関係する RSA 暗号が現実社会で利用され始めると研究目的として「正確な安全性評価」という錦の御旗が使えることから競争がさらに加速されてきた．**表 -1** に今世紀以降の（一般的な形の合成数の）素因数分解記録が更新されてきた歴史をあげる．

発表年	桁数（ビット数）	分解者
2002	158 (524)	Bonn 大
2003	160 (530)	Bonn 大
2003	174 (576)	Bonn 大
2005	176 (582)	NTT, 立教大, 富士通研
2005	200 (663)	Bonn 大
2010	232 (768)	EPFL, NTT, Bonn 大, INRIA, CWI

注）この表はそれぞれの分解が発表された当時，世界記録だったもののみを挙げているので，分解された数の大きさの順位とはなっていない．

表 -1　今世紀における素因数分解記録更新の歴史
（2010 年 6 月現在）

RSA 暗号が提案されたとき，発明者により RSA で暗号化された文章の解読に関する懸賞金問題が提出された．そこで使われた数は後に RSA-129 と呼ばれる 129 桁の合成数であり，1994 年にこの数を素因数分解することにより懸賞金問題が解かれた．その後，徐々に素因数分解記録が伸びており，中でもビット表現および素因数分解法に対して，きりの

† 本稿の内容は筆者一個人の見解であって所属機関の公式見解ではないことを付記する．

よい数が分解記録の一里塚と考えられ，それらの分解競争が繰り広げられてきた．特に，RSA-155（512ビット）の分解が 1999 年，kilo-bit SNFS（1000 ビットを超える合成数の特殊数体篩法（とくしゅすうたいふるいほう）による素因数分解）として M1039 $(=2^{1039}-1)$ が 2007 年に分解された[6] ことは記憶に新しい．その後，一般の形である 768 ビット合成数の素因数分解が研究者の目標となった．

2009 年 12 月 12 日（日本時間では 13 日，発表は 2010 年 1 月），私が参加するチームが RSA-768 と呼ばれる 768 ビットの合成数の素因数分解を完了した（表 -1 最終行）．今回の記録は 2005 年に Bonn 大のチームがうちたてた記録をおよそ 5 年ぶりに更新するものである．この記録は EPFL（瑞，Ecole Polytechnique Fédérale de Lausanne），NTT（日，Nippon Telegraph and Telephone），Bonn 大（独，die Universität Bonn），INRIA（仏，Institut National de Recherche en Informatique et en Automatique），CWI（蘭，Centrum voor Wiskunde en Informatica）の 5 カ国の機関の研究者の共同作業の成果である．

RSA-768 とは，RSA 社が RSA 暗号の安全性評価に貢献するとして挙げていた RSA Factoring Challenge にある合成数の 1 つであり，768 ビット（232 桁）である．RSA Factoring Challenge にある合成数は，実際に RSA 暗号で使われ得る形の合成数で，桁数が同じならば最も素因数分解が難しいと考えられている形である．Challenge が出された当初は懸賞金がかかっていたが，2007 年に，もうその役目は終えたとして Challenge は終了した．

❑ チーム編成の経緯

表 -1 から読み取れるように，2005 年までは Bonn 大が若干優位に立ちながらも，NTT らのチームと激しく競争を繰り広げていた[☆1]．その後，素因数分解研究の大御所である EPFL の A. K. Lenstra の仲立ちにより，Bonn 大の Kleinjung らとの共同研究を開始した．その結果，M1039 $(=2^{1039}-1)$ という特別な型に関する素因数分解の記録更新（kilo-bit SNFS）に繋がった．その後も共同研究は継続し，次の目標を一般的な型の素因数分解である RSA-768 に定め，研究を継続した．当初は，当時利用可能であった計算資源で行列処理が可能かどうか危ぶまれたものの，篩処理が終わるころにはなんとかなるだろうとの楽観視のもと分解作業を開始した．2008 年 10 月に INRIA 等が主催の素因数分解に関するワークショップが開催された．A. K. Lenstra らのもとに，チームへの参加希望が来ていた INRIA と CWI の関係者との打合せの結果，彼らも共同研究に加わることに合意した[☆2]．CWI は以前 A. K. Lenstra が在籍していたこともあり，1990 年代の世界記録更新にほぼすべてかかわっていたことがある素因数分解実験については歴史ある研究機関である．INRIA は最近の楕円曲線法での素因数分解の世界記録更新についてはそのほぼすべてにかかわる GMP-ECM というフリーソフトを開発している実力者揃いである．さらに，CADO 事業[☆3] の関係で数体篩法[4] に関する素因数分解も積極的に行い始めており，grid 5000 などの豊富な計算機資源にもアクセス可能であり心強い援軍であった．

❑ 素因数分解問題の性質

素因数分解は学校教育で習うように，N を $\lfloor\sqrt{N}\rfloor$[☆4] までの素数で順に割っていくことにより素因数分解できる．この方法を試行除算法（TD；trial division）と呼ぶ．この方法では最悪の場合に，$\sqrt{N}$ 以下の素数すべてについて割算を試すことになる．では，その $\sqrt{N}$ 以下の素数すべてをどう得るのだろうか．これは Eratosthenes の篩と呼ばれる非常に効率的な方法が知られている．たとえば次の手順で行われる．

(1) 1 ～$\lfloor\sqrt{N}\rfloor$までのマスを準備する．
(2) 1 を取り除く．
(3) 1 の次に一番小さな数を探し，○印をつける．

☆1 数週間程度の差で数体篩法の世界記録がとれなかった事例も複数ある．

☆2 正式な合意は書類処理の関係上，それから数カ月後になった．

☆3 ANR の出資による事業で，数体篩法による整数の素因数分解研究と実装を目的とする．

☆4 N の平方根を超えない最大の整数を表す．

これは 2 である．

(4) マスの 2 より大きな数に対して，すべての 2 の倍数に×印をつける．

(5) (3) に戻り，同様に繰り返す．つまり，次の繰返しでは，マス中のいかなる印もついていない数の最小のものは 3 であるので，3 に○印をつけ，3 より大きなすべての 3 の倍数に×印をつける．同様に 5, 7, 11, ... と繰り返してゆく．

といった，作業を$\sqrt{\sqrt{N}}$までの素数まで繰り返せば$\sqrt{N}$までのすべての素数が得られる．素数定理によりx以下の素数の数はおよそ$x/\log x$であることが知られているので，試行除算法による素因数分解は，最悪の場合，およそ$2\sqrt{N}/\log N$回の除算が必要となる．つまりNがたとえ 40 桁程度しかなくても，その辺のパソコンでは，たちうちできないことになる．

逆にかけた結果 40 桁になる 20 桁同士の乗算はどうだろう．これは非常に容易な問題である．実際，ちょっと根気のある読者であれば，コンピュータに頼らずとも筆算で 1 時間もかからずにできるのではないだろうか．

面白いことに素数判定は非常に易しい問題であることが知られている[☆5]．今回の記録となったのと同程度の大きさの整数が素数であることを示すにはその辺のパソコンで数秒もあれば十分である．

今日では，素因数分解問題については試行除算法よりかなり効率のよい方法が知られてはいるが，それでもまだ乗算，または素数判定よりははるかに時間がかかる方法しか知られていない．ただし，素因数分解問題の中には非常に簡単な問題があることに注意しよう．たとえば，与えられた整数が素数であれば，素因数分解は素数判定 1 回だけで済む．また，いくつかの小さな素数と 1 つの大きな素数の積となる場合は，小さな素数を試し割りで検出したあと，残りの整数を素数判定すれば素因数分解が終了するので易しい問題である．そのほか，既知の素因数分解法に対しては，上記 TD の場合にいくつかの小さな素数と 1 つの大きな素数の積となる合成数に対しては効果的であったように，得意となる型がある場合がある．以下では一般の場合の素因数分解問題とは，そのような場合を除いたものを指す．今回の分解対象となった RSA-768 もそのような一般の場合の合成数である．

さまざまな素因数分解法やその周辺アルゴリズムについては文献 2) が詳しいので参照されたい．

[☆5] 有効な合成数判定法として Miller-Rabin 法，素数判定法としては多項式時間ではないが実用的な APRT-CL 法や，漸近的には高速な AKS 法が知られている．

❑ 公開鍵暗号と素因数分解

皆さんは Web ブラウザの南京錠マークを見たことがあるのではないだろうか．これは TLS（もしくは SSL）と呼ばれる暗号通信が行われていることを表しており，(正しく使われている限りは) 安全な通信が行われていることを示している．ここで活躍するのが公開鍵暗号である．今日の暗号技術では第二次世界対戦中までに使われた暗号のように，方式そのものを非公開にする必要はなく，「鍵」と呼ばれる，暗号方式の記述に比べれば相当短い情報だけを秘密にすれば安全性が保たれるように設計されている．しかし，その方法でも通信相手と秘密を保ったままどのように鍵を共有すればよいかが問題であった．この問題を解決したのが公開鍵暗号である．公開鍵暗号では，暗号化用の鍵と復号用の鍵を別々に準備し，復号用の鍵は従来通り秘密に保つ必要があるが，なんと，驚くべきことに暗号化用の鍵は一般に公開してしまっても安全性が保たれるというように設計されている．この暗号化用の鍵を公開鍵，復号用の鍵を秘密鍵と呼ぶ．公開鍵暗号を用いて暗号通信を行う場合は，公開鍵と秘密鍵を生成した後，秘密鍵はそのまま秘密に保持し，公開鍵を相手方にそのまま送ればよい．このとき，盗聴されても秘密鍵に関する情報は洩れないように設計されているので，安全である．このように，公開鍵暗号は従来型の暗号に比べ，処理速度は遅いものの使い勝手が非常によく，今日の安全な通信を実現する上でなければならない方法である．その，公開鍵暗号の事実上の標準として RSA という方法がある．RSA 暗号の安全性は素因数分解問題の困難性と関係しており，実際，

RSA で利用される合成数が分解されると RSA が解読されてしまうことが知られている．現在，RSA 暗号の安全性評価に素因数分解記録の進展は欠くことができないものとなっている．

数体篩法

この章では数体篩法の概要について説明する．数体篩法（NFS；number field sieve）とは素因数分解法の 1 つであり，一般の場合の素因数分解について，既知の方法☆6 では，漸近的に最も高速な方法である．実際，表 -1 の分解はすべて数体篩法により作られた記録である．数体篩法の計算量は準指数を表す $L_N[s,c]$ $(=\exp((c+o(1))(\log N)^s(\log\log N)^{1-s}))$ という関数を用いて評価される．ここで $L_N[s,c]$ は，s=0 とおくと N の長さに関する多項式になり，s=1 とおくと指数関数となることから，$0<s<1$ とすることにより多項式と指数関数を結ぶ準指数となる．この関数を用い TD の最悪計算量を書き表すと $L_N[1,1/2]$ となり，数体篩法はいくつかの妥当と考えられている仮定のもと，平均計算量の上限が $L_N[1/3,(64/9)^{1/3}]$ と評価される☆7．ここで $o(1)$ は $N\to\infty$ のとき 0 に収束する関数である．具体的に与えられた N について，この $o(1)$ がどのような大きさとなるのかはよく分かっておらず，具体的な計算機実験の積み重ねにより評価されてきている．参考までに**図 -1** に $o(1)$=0 とおいた場合の $L_N[1,1/2]$ と $L_N[1/3,(64/9)^{1/3}]$ のグラフを示す．このグラフは両対数グラフであり，横軸を N，縦軸を L_N としている．このグラフから準指数関数は指数関数に比べてきわめてゆっくりと増加することが読みとれるだろう．なお，実際の素因数分解の計算量は先に述べたように $o(1)$ の挙動によることから，その値により相当大きく変化するので，このグラフから未分解の大きな具体的な大きさの合成数に対する素因数分解の計算量を推測してはならないことに注意されたい．

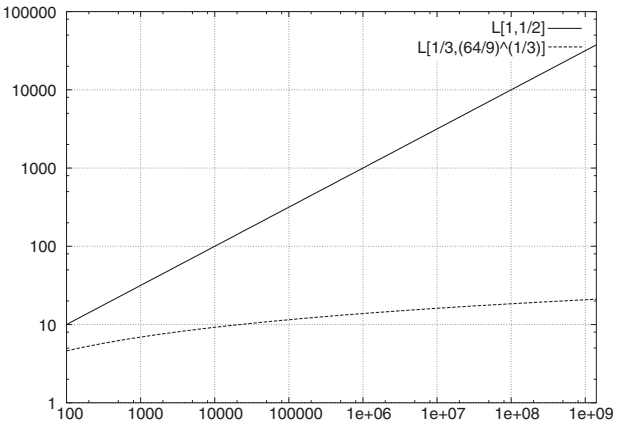

図 -1　準指数関数

いくつかの素因数分解法と同様，数体篩法は $x^2\equiv y^2 \pmod N$ となる非自明な x,y の組を求めることにより素因数分解が行われる☆8．そのような x が求まると最大公約数 $\gcd(x\pm y,N)$ を計算することにより N の因子が求まる．このような x,y の組はどのようにしたら求まるのであろうか．たとえば Dixon のランダム平方法では次のように行われる．

(1) x をランダムに発生させ，$r_x \leftarrow x^2 \bmod N$ を計算する．
(2) r_x をある閾値 B 以下で試し割りをし素因数分解する．
(3) r_x が完全に素因数分解されれば，それを記録．
(4) 必要数が集まるまで上記を繰り返す．

ここで十分に r_x が集まると，そのいくつかをかけ合わせることにより平方数となる．具体的な計算手順としては，p を素数として $r_x=\prod_{p<B} p^{e_p}$ と書かれているので，r_x の積は指数法則により実質的に素数 p のベキである e_p の和の計算になることを利用し，指数が偶数となる組合せを見つければよい．これは指数 e_p を並べたベクトルが定義する GF(2)☆9 上の連立方程式の非自明解を求めることに対応する．数体篩法は上記方法の考えを引き継ぎ，もう少し効率的に行われる．詳細については文献 4)

☆6 量子コンピュータが利用可能な場合はさらに高速な方法がある．
☆7 $L_N[s,c]$ の c の部分が $(64/9)^{1/3}$ より若干小さな改良法も知られているが現在の世界記録が達成されている辺りの N については，この改良法の方が遅いと考えられている．
☆8 $a\equiv b \pmod N$ とは $a-b$ が N で割り切れることを示す．整数を足したものや，かけたものの余りは，余りを足したものや，かけたものに一致することが知られており等号「=」と同様の感覚で合同記号「≡」が用いられる．
☆9 GF(2) とは {0, 1} のみからなる体．整数とほぼ同様に四則演算が定義できる．ただし，1+1=0 である．

を参照されたい．その，数体篩法はいくつかの大きなステップにより構成されている．以下にそれぞれのステップの概要を示す．

❑ 多項式選択

$f_0(M) \equiv f_1(M) \equiv 0 \pmod N$ となる互いに素な整数係数既約多項式 $f_0(X), f_1(X)$ と整数 M を探す．最適な多項式の次数の合計は N の大きさにより決まり，現在の世界記録程度の N の場合は 7 程度である．ここで選ばれた多項式により，後続のステップの計算量が大きく左右されるが，どの程度の時間をかけて多項式選択すればよいか，どの程度の品質の多項式を探し出せばよいかについては，残念ながら経験と勘に頼るしかないのが現状である．

❑ 篩

いわゆる Eratosthenes の篩と類似の処理を行う．$f_i(X)$ により計算される，N より小さく，しかしそこそこ大きな数に対して，小さな素数のみで割り切れる数を多数探す．この数は $f_0(X)$ と $f_1(X)$ の関係を表すことから「関係式 (relation)」と呼ばれ，このステップを「関係式収集ステップ」と呼ぶこともある．

このステップは数体篩法の計算量評価で支配項となり，実際に計算機実験を行っても最も計算量を要してきた．ただし，計算を多数の独立な計算に分割できることから，コンピュータの台数さえ確保できれば，比較的容易に計算可能である．

❑ 線形代数

理論的には，篩出力で得られた多数の関係式から GF(2) 上の巨大な行列を作り，その行列で定められる連立方程式の非自明解をいくつか求めるステップである．計算機実験的には，巨大な行列をある程度小さな行列に変換し，それが定める連立方程式の非自明解を求めるという，2 つのステップにより構成されることが多い．前者はフィルタリングと呼ばれ，篩出力の多数のデータを処理することが主目的である．後者を行列処理と呼び，純粋に行列を解くことが目的となる．ここで生成される行列は疎行列であるので，Gauss の消去法のような一般的な行列解法ではなく，共役勾配法のような行列の次元 n に対して $O(n^2)$ で計算できる反復法が用いられる．フィルタリングでは経験上，行列を疎に保ったまま 1/10 から 1/100 程度に小さくできるので，後続の計算が 100 倍から 10000 倍速くなることになり，計算機実験では必須の技術である．

線形代数ステップは，篩ステップと同様，計算量評価で支配項となるが，これまでの計算機実験では篩ステップより，同程度以下の計算量となっている．ただし，現在のところ，有効な並列計算法は知られておらず，分割した問題を担当するそれぞれの計算ノード間の密結合が必要であることから，1 カ所にある程度以上の計算能力を集めなければならない．また，密結合が必要なことから分かるように 1 計算ノードの故障は全体の計算を止めてしまうことになるので，将来の記録更新のためには，計算機運用の面を忘れることはできない．ただし，行列処理は分散計算困難ではあるが，不可能というわけではない．block Wiedemann 法を用い，4 ～ 8 クラスタ程度への分割は可能である．篩処理のようにさらに大規模な分散計算を可能とするためには，さらなる研究が必要である．

❑ 平方根

線形代数ステップから出力された非自明解により定められる代数的数の平方根を計算する[☆10]．ここまでのステップは，ほとんど数論の知識がなくても理解し，プログラムを作成可能であるが，このステップだけは避けることができない．計算量はほとんど要しない．また，計算された平方根をもとに，最大公約数計算をすることにより，因子が求まる．

計算機運用

この章では RSA-768 分解で行った計算機運用について実際に行ったこと，起きたことを紹介する．

☆10 今回の場合，代数的数とは $f_i(X)$ の $\mathbf{C}$ 上の根の冪乗 (べきじょう) の有理数による線形結合である．

❑ 篩処理

篩処理の分散計算については，Lenstra らによりすでに 1989 年には「Factoring by Electronic Mail」という論文[5] が発表されるなど，20 年以上昔から疎結合ネットワークによる分散計算が行われてきた．今回も，参加組織間で提供可能計算資源を申告し，担当篩範囲を電子メールで申告しながら計算を進めた．

篩処理を行った結果の関係式については，EPFL で一括して収集したが，万が一のディスク破損等を考え，コピーを参加組織でも持つようにした．データの送付には Internet を通じ scp などを利用した．NTT 側は B フレッツを利用したが，のぼり回線を継続して利用するので，ISP の 1 日の利用量の制限にかからないよう帯域制限をかけて転送した．逆に下りについては EPFL の学内の利用制限を越えないよう，こちらも帯域制限を行いながらダウンロードした．最終的にダウンロードした関係式ファイルは gzip 圧縮で 5TB 程度であった．

このようなやりかたで，電子メールも頻繁なときでも 1 日に 1 ～ 2 往復，そうでないときは 1 カ月以上も音信不通というようなやりかたでも十分に機能し，複数組織間で篩領域が重なるといったような無駄が出ることもなく無事終了した．

なお，一度 EPFL の RAID ディスクが故障し，一部，復旧できないデータが出たが，これも無事，別組織のコピーから復旧でき，若干の人的時間の損失で済んだ．

❑ 行列処理

今回の分解では，M1039 分解のときに 4 並列に分解して実行してきたのと同様，8 つの計算に分け，それぞれを密結合された PC クラスタで計算した．ただし，分けられた計算は完全に独立とはいかず，計算およそ半ばで，いったん，途中計算結果を 1 カ所に集め，そう長くはないが非常にメモリを必要とする計算を行い，また，その結果を分配し，また，最後に 1 カ所に結果を集める必要があった．今回は，1 カ所に集めなければならない計算を

図 -2　NTT 設置クラスタ

EPFL で行い，その他の計算を EPFL，INRIA，NTT で行った．**図 -2** に NTT で利用したクラスタを示す．8 つに分けた計算は，どれもほぼ同じ計算量となるが，今回，かけた計算量がある程度はばらついたとしても，その結果が無駄にならずに使える手法を新たに利用した．とはいえ，計算能力の差があるので，必要に応じ，他所の計算を他に移行するといったことも行った．そのためには数 GB ある中間計算結果を送る必要があるので，半日から 1 日程度の転送時間を要した．**図 -3** に実際に利用した計算内容の割当表を示す．ここで，色の濃い方から順に EPFL, INRIA, NTT の担当計算を表す．

INRIA では grid 5000 という相当大きなクラスタ群を利用可能であったが，多数の独立した，さらに細切れの時間のみ利用可能という状況だったので，適切な時間で中間計算結果を生成し，また，その中間計算結果を利用して別のクラスタで計算することを再開するということを繰り返し実行した．支援スクリプトが作成されていたとはいえ，ローカルディスクがあまり使えないなどの制約が多い中，INRIA チームの貢献は派手ではないが大きな力となった．

❑ 故障

篩処理については，全体の集計サーバ，また問題の分配サーバはともかく，各計算ノードについては，まったく独立に動作するので特に問題はない．また，各計算ノードから上がってきた関係式ファイルの整合性は容易に検査可能なので，計算ミスがあった

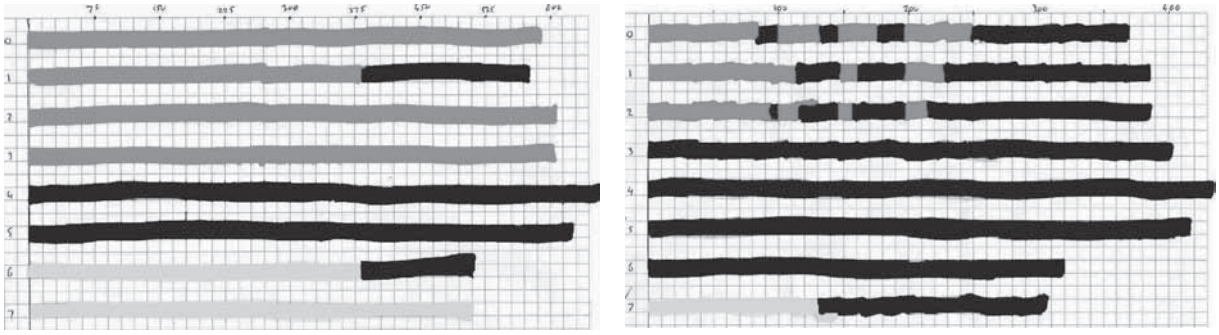

図 -3 計算割当表

としても容易に排除できる．今回，NTT では半数近くの計算ノードでメモリ故障☆11 にみまわれたが，単にその計算ノードを修理し，計算再開するだけで特に問題はなかった．

行列計算については，計算に参加している 1 ノードが停止するだけで，そのクラスタの全体計算が停止してしまうので，問題はやっかいである．今回 NTT で出会った障害はメモリとマザーボードの交換といった軽微な修理であり，幸いにして数回の対応ですんだ．また，交換まで計算できないのはもったいないので，計算 110 ノード以外に 3 台あった予備を VLAN で交換ノードと論理ネットワーク的にも，また物理ネットワーク的にもほぼ同じ位置に配置し，計算を続行した．この 3 ノードは物理的接続変更をしない場合には全体の 90% 程度の計算ノードにしか置き換えられなかったが，幸いにして，置き換え可能な計算ノードしか故障しなかった．

すでに故障に関する問題については，これまでの世界記録にかかわるような大きな計算を行ったときに多数経験しているので，ノウハウの蓄積があり，今回の計算では大きな問題はなかった．これまでの大規模な素因数分解では，メモリ，ハードディスク，グラフィクスカード，マザーボード，イーサネットスイッチ，電源，CPU，というありとあらゆるものが壊れた経験があり，中でも 1 カ月に一度くらいの頻度でおかしな結果を返すメモリと CPU には泣かされてきているので，検算も適切なタイミングで行っている．対処法としては，軒並であるが，篩計算については生成された多数の関係式ファイルのバックアップを作成し，行列計算では，適切な時間ごとに計算再開できるよう中間計算結果ファイルを作成，また，検算を行った．

☆11 さすがに半数近くの計算ノードでの故障はおかしいということで購入元に調査を求めたところ，納入したクラスタ 113 台のうち，およそ半数のノードで使われていたメモリがロット不良の可能性があるとのことで，対象ノードのメモリを全交換していただいた．ただし，他の顧客で同じような症状を示していたサーバはないとのこと．篩計算はメモリを酷使するので，購入後約 2 年という短い期間で問題が顕在化したものと思われる．

結果の詳細

この章では，計算結果，および計算量について簡単に説明する．

今回のおよそ 3 年に渡る計算の結果，

1230186684530117755130494958384962720772
8535695953347921973224521517264005072636
5751874520219978646938995647494277406384
5925192557326303453731548268507917026122
1429134616704292143116022212404792747377
9408066535141959745985690214341 3

=

3347807169895689878604416984821269081770
4794983713768568912431388982883793878002
2876147116525317430877378144679994 89

×

3674604366679959042824463379962795263227
9158164343087642676032283815739666511279
2333734171433968102700927987363089 17

と分解された．

要した計算量は，およそ 1700 core・年であった．**表 -2** に処理ごとの内訳を示す．ここでの計算量見

処理	core・年	期間
多項式選択	40	2005 年夏・2007 年初頭
篩	1500	2007 年 8 月～ 2009 年 4 月
フィルタリング	0.7	2009 年 2 月～ 2009 年 8 月
線形代数	155	2009 年 8 月～ 2009 年 12 月
平方根	0.003	2009 年 12 月

表 -2　計算量内訳

積りは多数の計算機を利用したことからかなり大雑把であるが，1 core はおよそ Opteron 2.2GHz もしくは Core2 2.66GHz である．期間については，途中に計算停止期間もあり，また，途中での利用計算機台数の増減，最終的には使わなかった計算も含むことに注意されたい．

多項式選択は，2005 年夏に Bonn 大でおよそ 20 core・年で見つかったものを利用した．その後，2007 年初頭に EPFL でさらに 20 core・年をかけたが以前に見つけたものより，明らかによいものは見つけられなかった．

篩処理は 2007 年夏に開始し，2009 年春に終了した．計算の多くは最後の半年に行われた．関係式は当初予定していた 60×10^9 個を若干超える 64334489730 個集められた．後続のフィルタリングの結果，実はこの半数程度の関係式で数学的には線形代数処理が実行可能なことが判明している．ただし，その場合に，より線形代数処理が重くなるため，我々の準備できた計算資源で計算可能だったかどうかは不明である．

フィルタリングは 10TB のハードディスクを用い，8 コアの計算機で行われた．この処理では，壊れたファイルの除去，誤った計算結果の除去，利用した篩処理のアルゴリズム上避けられない重複した関係式の除去，なども並行して行った．行数(列数)と非ゼロ要素数が異なるいくつかの行列を生成し，EPFL および NTT で利用可能な計算機に対して最も適した行列を選んだ．

線形代数では非ゼロ要素数が 27797115920 である 192796550 × 192795550 の GF(2) 上の行列で定義される連立方程式の非自明解を求めた．アルゴリズムとしては block 長が 8 × 64 となる block Wiedemann 法を用いた．前半のスカラ積計算では最大 8 並列の計算を行った．途中の Berlekamp-Massey ステップでは，1TB 程度のメモリを要する計算が必要となり，ハードディスクへのスワップも利用した．最後の多項式評価計算では，最初のスカラ積計算で保存していた途中計算結果を利用し，可能な限り並列に計算した．

最後の平方根ステップでは，多項式の判別式の因子に関係するバグがあり，その修正に数日かかったものの，それ以外の問題はなく，無事，因子を発見できた．

所感

この章では，結果の意義，および 1024 ビット素因数分解について簡単にふれる．

今回の RSA-768 の分解は，RSA-512 の分解，kilo-bit SNFS に引き続く，素因数分解計算の一里塚である．計算時間には 3 年程度と長い時間をかけたものの，これまでに培ってきたノウハウのあるメンバでの作業であり，大きな困難もなく分解に至った．今回の結果の技術詳細は論文[3]を参照されたい．

ここで話を終わりたいところだが，1024 ビット素因数分解についての話題を避けるわけにはいかないだろう．現在，RSA 暗号の利用で最も利用されているのが 1024 ビット合成数の素因数分解の困難性に安全性が基づくものである．現在，TLS（もしくは SSL）サイトの 80% 以上で利用されているようである．つまり，1024 ビット合成数の計算困難性の評価は，現在利用されている RSA 暗号の利用について最も重要な課題である．すでに，CRYPTREC Report 2006 などで示唆されているように，現在ではスーパーコンピュータを利用すれば 1024 ビット素因数分解は可能な領域に入っていると考えられている．

一般の RSA 暗号利用者が何をすべきかについては，その人の価値判断に委ねたい．以下には，その判断に必要な情報を可能な範囲で書きたいと思う．

- ある合成数が分解されたからといって，利用して

いる秘密鍵が知られてしまったというわけではない.

- 現在世界最高速のスーパーコンピュータを 1 年程度利用できれば素因数分解可能と考えられる.
- $o(1)$ 項を無視した素因数分解計算量の評価式を用いた荒い評価では,RSA-768 分解の 800 ～ 1200 倍程度の計算量を要する.
- RSA-768 分解に用いた技術から,若干の数体篩法の改良がある.
- Brent の具体的に素因数分解された記録の推移の予測曲線[1] に従えば 5 ～ 10 年程度で具体的な分解記録が出る.

また,暗号化対象の情報は具体的な価値を持っていることも想定される.その価値を減じさせる要因は,利用した公開鍵に用いられる合成数の素因数分解だけに限らないかもしれないことにも注意をしたい.上記のように安全性は安全・危険と 0, 1 で判断できるものではなく,事情により異なる.個人的意見としては,新規の 1024 ビット利用はやめ,近い将来の分解報告が来ても慌てず,それに耐えられるよう準備をしてほしい.

参考文献

1) Brent, R. P. : Recent Progress and Prospects for Integer Factorisation Algorithms, Computing and Combinatorics: 6th Annual International Conference, COCOON 2000 (Du, D.-Z., Eades, P., Estivill-Castro, V., Lin, X. and Sharma, A., eds.), Lecture Notes in Computer Science, Vol.1858, Berlin, Heidelberg, New York, Springer-Verlag, pp.3-22 (2000).
2) Crandall, R. and Pomerance, C. : Prime Numbers ― A Computational Perspective, Springer (2001).
3) Kleinjung, T., Aoki, K., Franke, J., Lenstra, A. K., Thomé, E., Bos, J.W., Gaudry, P., Kruppa, A., Montgomery, P.L., Osvik, D.A., te Riele, H., Timofeev, A. and Zimmermann, P. : Factorization of a 768-bit RSA Modulus, Advances in Cryptology ― CRYPTO 2010 (Rabin, T., ed.), Lecture Notes in Computer Science, Vol.6223, Berlin, Heidelberg, Springer-Verlag, pp.333-350 (2010). to appear.
4) Lenstra, A. K. and Lenstra, Jr., H.W.(eds.) : The Development of the Number Field Sieve, Lecture Notes in Mathematics, Vol.1554, Springer-Verlag, Berlin, Heidelberg (1993).
5) Lenstra, A. K. and Manasse, M. S. : Factoring by Electronic Mail, Advances in Cryptology ― EUROCRYPT'89 (Quisquater, J.-J. and Vandewalle, J., eds.), Lecture Notes in Computer Science, Vol.434, Springer-Verlag, Berlin, Heidelberg, New York, pp.355-371 (1990).
6) 青木和麻呂：素因数分解の世界記録はいかに作られたか，電子情報通信学会誌，Vol.91, No.6, pp.462-468 (2008).

(平成 22 年 6 月 3 日受付)

青木和麻呂 aoki.kazumaro@lab.ntt.co.jp

昭和 44 年生.平成 7 年早稲田大学大学院理工学研究科修士課程修了.同年日本電信電話(株)入社.暗号の安全性評価研究に従事.平成 13 年通信・放送機構(現情報通信研究機構)に出向.平成 18 年度から平成 20 年度に早稲田大学基幹理工学研究科非常勤講師.平成 20 年度に名古屋大学客員教員.平成 20 年末から墺国 TU Graz 客員教授.現在 NTT 情報流通プラットフォーム研究所主任研究員.博士(理学).SCIS'95 論文賞,SCIS'96 論文賞,電子情報通信学会平成 9 年度学術奨励賞,平成 17 年度業績賞,IWSEC 2007 BEST PAPER AWARD 受賞.電子情報通信学会,IACR 各会員.